L8: Entry 4 of 6 File: USPT Oct 15, 1996

DOCUMENT-IDENTIFIER: US 5566332 A
TITLE: Method and combination for minimizing data conversions when data is transferred between a first database storing data in a first format and a second database storing data in a second format

Brief Summary Text (2):
This application includes material which overlaps material in pending U.S. patent application Ser. No. 499,114, filed Mar. 26, 1990, entitled "METHOD AND APPARATUS FOR DESCRIBING DATA TO BE EXCHANGED BETWEEN PROCESSES", inventors: Richard Demers, et al., now U.S. Pat. No. 5,278,978, and U.S. patent application Ser. No. 500,032, filed Mar. 27, 1990, entitled "QUERY LANGUAGE EXECUTION ON HETEROGENEOUS DATABASE SERVERS", inventors: John G. Adair, et al. now, U.S. Pat. No. 5,257,366.

Brief Summary Text (9):
The machines upon which the DBMSs of a heterogeneous database system run all represent information in different internal formats. For example, numeric information on PS/2 machines is stored with the bytes in low order to high order sequence. On other machines, such information may be stored in high order to low order sequence. For floating point information, there are IEEE floating point machines and hexadecimal floating point machines. Character information is processed in many different code representations, the choice of which reflects historical or cultural roots.

Brief Summary Text (13):
Provision is made in the prior art for solving the problems of machine and system incompatibility in a distributed, heterogeneous database system. Three solutions are of interest.

Brief Summary Text (14):
The earliest solution may be termed "application beware". This solution usually starts as a connection between identical database systems which grows over time to incorporate some machines which differ slightly from the original. In these solutions, there is no way for the system to automatically handle the differences, with the result that the application program was given this responsibility. If access to heterogeneous databases was needed badly enough, the application was written to make any necessary accommodations.

Brief Summary Text (17):
Thus, there is an evident need in distributed, heterogeneous database systems to support effective and accurate exchange of data, while reducing the number of conversions, and the communications overhead. It is also desirable to perform any needed conversion at the site where the data to be converted will be processed.

Brief Summary Text (20):
To minimize the number of conversions, a method is provided whereby no data is converted until it is needed in a specific format for processing. This method allows each system to always send all data in its preferred (native) format, or the current format of the data if the data is just passing through an intermediate system without being processed. When data is received, it is converted to the native format of the system only when it has reached its final destination (in being passed to the application, or stored in the database), or when it actually needs to be processed by an intermediate system.

Brief Summary Text (21):
According to the invention, a database in a distributed, heterogeneous database system contains predefined descriptions of all machine environments in the system (machine descriptors) and

predefined descriptions of database language structures (system descriptors) for each DBMS with which it can perform data exchange. When a database operation begins which requires two heterogeneous databases to conduct an information exchange, a communication link is established between them. Next, each DBMS identifies its machine and system descriptors to the other. This establishes a data context and is done only once for the life of the communication link. Once established, requests, responses and data can be sent or received. When any data is sent, it is sent in the native format of the sender. Specific descriptions of the data precede the data itself and refer to the machine and system descriptors earlier identified. When the data is received, information contained in the specific descriptions enable these descriptions to be referenced to the machine and language descriptors and passed off to a conversion process in the receiver. Taken together, the specific descriptions, and the machine and system descriptors which they reference, precisely characterize the environment where the data originated and establish, at the receiver, a context for accurate conversion of the data into a format which is native to the machine/system combination of the receiving site.

Detailed Description Text (5):
Furthermore, there are many SQL identified standard control blocks, such as the SQL communication area (SQLCA) which is defined in terms of SQL types. The SQLCAs in the machine environments defined above are not identical. This invention formalizes a method and means for exchanging information between heterogeneous DBMSs about machine characteristics and DBMS language structures such that little or no descriptive information must be exchanged at run-time in order to convert data to a native form. Additionally, when database sites match, even though the data may have been forwarded through intermediate DBMS systems which do not match, no conversions are performed at all, thus preserving completely the integrity of the data being exchanged. When these sites do not match, data conversions are performed close to the ultimate point of use where errors introduced by imperfect conversions can be dealt with according to the needs of the requesting application.

Detailed Description Text (15):
The context for conversion also requires information describing characteristics of the language in which the DBMS sending the information is written. In this description it is assumed that all of the DBMS sites are written in one or another version of an SQL-based language. In order to accommodate non-identical representations of control information produced by these varying versions, a system level language characteristic descriptor must be available to the converting machine in order to convert the language-specific portions of the user data. In FIG. 1A, the language-specific portions are the communication areas. In order to provide this system-level information about the language characteristics, the converting machine is provided a system-level information unit descriptor. In FIG. 1B, the system-level descriptor for the personal computer language environment is indicated by reference numeral 45.

Detailed Description Text (18):
Thus, in FIG. 1B, assuming that the mainframe machine 50 is sending data, its machine-level and system-level descriptors 54 and 55 must be made available to the requesting personal computer machine 40 and linked to application level descriptors 57 in order to convert the user data of FIG. 1A from the representation 20 to the representation 10. Relatedly, the descriptors 54, 55 and 57 must be made available to the machine 40. Similarly, for data transferred in the opposite direction, that is, from the personal computer machine 40 to the mainframe machine 50, the descriptors 44, 45 and 47 would have to be made available to the mainframe machine 50 in order to convert user data 10 into the form of user data 20 in FIG. 1A.

Detailed Description Text (19):
In the practice of this invention, it is asserted that, during the implementation of each DBMS site which participates in a distributed, heterogeneous database system, a decision is made as to which specific machine representations will be supported at the site. It is asserted that the "native" representation of the machine on which the site's DBMS will run is supported. That is, if data from another DBMS identical to the receiving site's DBMS is received, the representation is "native" and requires no conversion. Additionally, other machine types will be supported as partners. Generally, the "non-native" representations of data types will be converted to "native" ones at the receiving site before processing. In the preferred embodiment, at each site, there is maintained a list of acceptable machine partner types at each site. Thus, at the site of the personal computer machine 40, a list of acceptable machine types includes the machine and code page corresponding to identifiers 51 and 52 in the machine 50. The list indexes to a set of descriptors which include the machine-level and system-level

descriptors for all acceptable system partners. Thus, if the personal computer machine 40 receives simply identification of the machine and language present at the site 50, these identifications can be brought to the list which will index to the necessary descriptors in the list of descriptors, the indexed descriptors corresponding to the descriptors 54 and 55.

Detailed Description Text (22):
Upon receipt, the receiver uses the descriptors obtained in response to the machine and system-level identifier sent by the machine 50 and the references contained in the application level descriptor 57 to convert the user data 20 from the server's representations to the receiver's representations for subsequent processing.

Detailed Description Text (39):
Refer now to FIG. 4 for an understanding of the structure and content of a user data descriptor and how its linkages with machine- and system-level descriptors provide a complete context which enables a receiver DBMS to understand and convert user data from non-native to native form. The user data descriptor 200 provides a system-level and machine-level-independent way to describe user data. It is contemplated that this descriptor would not be built until run time, since, particulars of user data aren't known until after installation of a DBMS and creation and population of the relational tables in the DBMS, and until receipt of a particular request for data.

Detailed Description Text (45):
FIG. 5 illustrates a procedure for establishing a conversion context using the data objects whose structures and functions have been described above. The procedure includes generic commands for requesting and completing a communication connection and for requesting and providing data. The procedure of FIG. 5 is illustrated as a simplified sequence which shows only the parameters which must, of necessity, be exchanged for transferring data between requesting and serving machines and establishing the translation context in the receiving machine for conversion of the data. It is reiterated that the procedure of the invention is not concerned with how data is converted, but rather with where the data is converted. Particularly, the invention enables the receiver of the data to convert the data for establishing particular values for those parameters.

Detailed Description Text (46):
The structure of FIG. 5 places all of the requesting machine actions in the left-hand side of the drawing and all of the serving machine actions in the right-hand side. Thus, in step 240 a user at the requesting machine first issues a request for data which is maintained by a DBMS in the serving machine. Next, in step 252 the requesting machine executes a REQUEST-CONNECTION command sequence directed to the serving machine. The REQUEST-CONNECTION sequence includes parameters to identify the requesting machine, its character code, and its language level. In the example, the requesting machine is identified as a DBMS executing on a PS/2 personal computer, utilizing a specific code page (437) for character coding, and operating at a specific DBMS language level (SQLAM3). The parameters in the REQUEST-CONNECTION step are no more than identifiers. Next, in step 254 the serving machine receives the connection request, validates the machine- and system-level identifiers, and uses these identifications to index to machine- and system-level descriptors. Assuming that the serving machine recognizes and possesses the machine- and system-level descriptors identified in the connection request, it executes a COMPLETE-CONNECTION in step 256 including parameters which provide machine- and system-level identification to the requesting machine. In step 258, the COMPLETE-CONNECTION response received from the serving machine is validated in the requesting machine and the identifiers are used to index the appropriate the machine- and system-level descriptors. For example, in the example of FIG. 5, the requesting machine would index to the machine and system descriptors 70 and 100 illustrated in FIG. 2B, 3B, and 4A. Next, in step 260, the requesting machine assembles a data request and sends it in step 263 as a REQUEST-DATA command to the serving machine. This REQUEST.sub.-- DATA optionally includes an input data descriptor (built in step 260) with optional input data, if appropriate for the SQL to be executed. In step 265, the serving machine receives the REQUEST-DATA command, including optional input data and descriptor. If input data is present, a conversion process is called in step 267. The serving machine responds to the REQUEST.sub.-- DATA command in step 269 by obtaining the requested user data and building a user data descriptor such as the descriptor 200 illustrated in FIG. 4A, and executes a PROVIDE-DATA command in step 270 by sending the user-data descriptor and the user data as, for example, the first row in the user data 20 in FIGS. 1B and 4B. The requesting machine then, in step 272, receives the transmitted descriptor and user data and calls a

conversion process to convert the data. According to the needs of a particular implementation, the bindings between the user data descriptor and the indexed machine and system descriptors can be done in step 272 or left for the called conversion process. In step 274, a second REQUEST-DATA command is sent, its parameters are processed, with the server, in step 276, obtaining and returning the requested user data with another PROVIDE.sub.-- DATA command (step 278). The requesting machine again, in step 279, processes the data according to the user data descriptor received with the first PROVIDE.sub.-- DATA command, and to the system level, machine type, and code page values received during the connection portion of the process. Now if the receiver requires more data, it issues another REQUEST-DATA command, with the server returning the data with another PROVIDE-DATA command, and so on.

Detailed Description Text (50):
Having verified that the receiver's environment can be handled, the application manager 318 constructs a COMPLETE-CONNECTION command as illustrated in FIG. 5. The command states the serving machine's characteristics in the MACH code page parameters and the level of system function which the serving machine will support in the DBMS parameter. It is contemplated in the invention that the connection steps may repeat in order to support negotiation to a different system level than that originally requested. Next, the COMPLETE-CONNECTION response is returned to the requesting machine 280 by sending it on 328 to the agent 314 which communicates it on the communication link 312 to the receiver agent 310. The receiver agent 310 sends the COMPLETE-CONNECTION command and parameters at 330 to the application manager 306. In a manner similar to the manager 318, the manager 306 accesses the system and machine-level descriptors which describe the server machine and system characteristics from the dictionaries 342 and 346. Having validated that connection has been established, and having obtained the machine- and system-level descriptors at both ends of the connection, processing continues. The requester's application manager 306 constructs a REQUEST-DATA command corresponding to the OPEN request from the application program 300. This may include an optimal input data descriptor and input data. This command is sent to the server's application manager 318 via 308, 310, 312, 314 and 316. The application manager 318 recognizes a "first request", processes any input data descriptor, converts any input data, and issues an OPEN command 332 to the local DBMS 334, which returns the status in the form of an SQL communication area 336. Next, the manager 318 issues a request on 332 to determine the format of the answer set. The DBMS 334 builds an SQL data area describing the rows which will be returned. This data area is returned on 336 to the server application manager 318 which constructs the user data descriptor for the data. Assuming that data is buffered into the manager 318 and that there is room in the buffer, the server's application manager 318 issues an SQL FETCH request on 332 to the DBMS 334 which returns a first row of data on 336. (It is assumed that the data to be returned corresponds to the user data 20 in FIGS. 1A and 4B.) Next, the server application manager 318 places the data in a reply buffer and issues a PROVIDE-DATA command to send the data to the requester's application manager 306 via 328, 314, 312, 310, and 330. It is contemplated, but not required, that the server's application manager 318 may read ahead to fill buffers in anticipation of the next REQUEST-DATA command.

Detailed Description Text (51):
Upon receiving data with the first PROVIDE-DATA command, the receiver's application manager 306 processes the user data descriptor to verify correctness and to prepare to convert data subsequently received. Assuming no errors are found, the application manager 306 then sends the result of the OPEN back to the application program 300 on 338. From now on, the receiver's application manager 306 may, but is not required to, read ahead requesting additional buffers frown the server's application manager 318 in anticipation of FETCH requests.

Detailed Description Text (52):
Having successfully opened a Cursor, the application program 300 issues an SQL FETCH which is processed by the receiver's application manager 306. At this time, the receiver's application manager 306 has a row of data to return, and converts the data and returns it on 338 to the application program 300. The application 300 processes the data and SQL communication area received and subsequently issues another FETCH command on 304 to obtain another row of data. The requester's application manager 306 having no data to satisfy this request sends another REQUEST-DATA command to the server's application manager 318 via 308, 310, 312, 314, and 316. The server's application manager 318 having read ahead, has a buffer containing the last two rows of data. These rows are sent with the PROVIDE-DATA command to the requester's application manager 306 via 328, 314, 312, 310, and 330. The application manager 306 then converts the first row of the buffer using the previously constructed conversion descriptors and passes it

on 338 to the application 300.

Detailed Description Text (53):
To complete the example, the application 300 issues another SQL FETCH request on 304 to request the last row of the answer set. The receiver's application manager 306 converts this last row of data and returns it to the application.

Detailed Description Text (64):
A data converter is not illustrated as an element in any of the figures, it being understood, that data format conversion is well-known in the art. For example, U.S. Pat. No. 4,559,614 of Peck et al., assigned to the Assignee of this application, describes in detail how conversion from a first to a second internal code format is accomplished for data transmitted between two dissimilar computer systems. To the extent necessary, this patent is incorporated herein by reference.

Current US Original Classification (1):
707/101

Other Reference Publication (4):
Strevell et al., "High Speed Transformation of Primitive Data Types In a Heterogeneous Distributed Computer System," Distributed Computing Systems 1988 Internat. Conf. pp. 41-46.

CLAIMS:

1. A combination for establishing a data conversion context at a first computer system which receives data from a second computer system, wherein the second computer system stores and processes data in a format different than the first computer system, the combination comprising:

(a) descriptors in the first computer system defining machine and language characteristics of a plurality of computer systems;

(b) means for obtaining from said descriptors, first and second descriptors which respectively define machine and language characteristics for said second computer system;

(c) means for receiving data from the second computer system; and

(d) means for combining the first and second descriptors with data descriptors in the received data which describe characteristics of data native to the second computer system to produce a context for converting the received data to data which is native to the first computer system.

2. A method for use by a user processor in making data requests to a server processor which stores and processes data in a format different than the user processor, comprising the user processor executed steps of:

storing a dictionary of processor descriptors;

creating a communication link to a server processor by:

communicating to the server processor an identifier denoting the user processor; and

receiving from the server processor an identifier denoting the server processor;

sending a request for data to the server processor in the user processor data format, the request including a descriptor of the user processor data format;

receiving from the server processor data responsive to the request for data, the data being in the server processor data format, and being accompanied by a descriptor of the server processor data format; and

using the identifier denoting the server processor and the descriptor of the server processor data format, converting the data to the user processor data format.

5. The method of claim 4, wherein:

the storing step includes storing a plurality of server processor machine descriptors at the user processor;

and the step of converting includes:

obtaining a server processor machine descriptor from the dictionary in response to the identifier denoting the server processor; and

in response to the server processor machine descriptor, converting bit representations in the data received from the server processor to bit representations used by the user processor.

7. The method of claim 6, wherein:

the storing step includes storing a plurality of computer language descriptors in the dictionary; and

the step of converting includes:

obtaining a computer language descriptor from the dictionary in response to the identifier denoting the server processor; and

in response to the computer language descriptor, converting control fields in the data received from the server processor to control fields of a computer language used by the user processor.

9. The method of claim 8, wherein:

the storing step includes storing a plurality of server processor code page descriptors in the dictionary;

and, the step of converting includes:

obtaining a server processor code page descriptor from the dictionary in response to the identifier denoting the server processor; and

in response to the code page descriptor, converting character representations in the data received from the server processor to character representations used by the user processor.

10. The method of claim 8, wherein:

the storing step includes storing a plurality of server processor code page descriptors in the dictionary;

and, the step of converting includes:

obtaining a code page descriptor from the dictionary in response to the identifier denoting the server processor; and

in response to the code page descriptor, converting control function representations in the data received from the server processor to control function representations used by the user processor.

11. A combination for use by a user processor in making data requests to a server processor which stores and processes data in a format different than the user processor, the combination comprising, in the user processor:

means for creating a communication link to a server processor by:

communicating to the server processor an identifier denoting the user processor; and

receiving from the server processor an identifier denoting the server processor;

a dictionary of server processor descriptors;

means for sending a request for data to the server processor on the communication link in the user processor data format, the request including a descriptor of the user processor data format;

means for receiving from the server processor data responsive to the request for data, the data being in the server processor data format and being accompanied by a descriptor of the server processor data format; and

means for converting the data to the user processor data format in response to the identifier denoting the server processor and the descriptor of the server processor data format.

14. The combination of claim 13, wherein:

the server processor descriptors include a plurality of server processor machine descriptors;

and, the means for converting includes:

means for obtaining a server processor machine descriptor from the dictionary in response to the identifier denoting the server processor; and

means for converting bit representations in the data received from the server processor to bit representations used by the user processor in response to the server processor machine descriptor.

16. The combination of claim 15, wherein:

the server processor descriptors include a plurality of computer language descriptors;

and, the means for converting includes:

means for obtaining a computer language descriptor from the dictionary in response to the identifier denoting the server processor; and

means for converting control fields in the data received from the server processor to control fields of a computer language used by the user processor.

18. The combination of claim 17, wherein:

the server processor descriptors include a plurality of server processor code page descriptors;

and, the means for converting includes:

means for obtaining a server processor code page descriptor from the dictionary in response to the identifier denoting the server processor; and

means for converting character representations in the data received from the server processor to character representations used by the user processor in response to the code page descriptor.

19. The combination of claim 17, wherein:

the server processor descriptors include a plurality of code page descriptors;

and, the means for converting includes:

means for obtaining a code page descriptor from the dictionary in response to the identifier denoting the server processor; and

means for converting control function representations in the data received from the server processor to control function representations used by the user processor in response to the

code page descriptor.

Previous Doc Next Doc Go to Doc#

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 6 of 6 returned.

☐ 1. Document ID: US 6754679 B2

L8: Entry 1 of 6 File: USPT Jun 22, 2004

US-PAT-NO: 6754679
DOCUMENT-IDENTIFIER: US 6754679 B2

TITLE: Computer system with a plurality of database management systems

DATE-ISSUED: June 22, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Oheda; Takashi	Tokyo			JP

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Hitachi, Ltd.	Tokyo			JP	03

APPL-NO: 09/ 823640 [PALM]
DATE FILED: March 30, 2001

FOREIGN-APPL-PRIORITY-DATA:

COUNTRY	APPL-NO	APPL-DATE
JP	2000-115782	April 11, 2000

INT-CL: [07] G06 F 17/30

US-CL-ISSUED: 707/201; 707/2, 707/10, 707/102, 707/205, 709/232, 709/249, 709/313
US-CL-CURRENT: 707/201; 707/10, 707/102, 707/2, 707/205, 709/232, 709/249, 719/313

FIELD-OF-SEARCH: 707/102, 707/104, 707/9, 707/10, 707/201, 707/200, 707/2, 707/205, 709/313, 709/249, 709/103, 709/104, 709/219, 709/232

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4464223	August 1984	Gorin	
4585516	April 1986	Corn et al.	
5006978	April 1991	Neches	709/102
5457793	October 1995	Elko et al.	707/205
5479656	December 1995	Rawlings, III	707/200
5630067	May 1997	Kindell et al.	709/231
5649168	July 1997	Huang et al.	703/23

5884028	March 1999	Kindell et al.	709/234
6122630	September 2000	Strickler et al.	707/8
6167405	December 2000	Rosensteel et al.	707/102
6509828	January 2003	Bolavage et al.	340/10.1
2002/0029224	March 2002	Carlsson	707/104.1

OTHER PUBLICATIONS

Semiconductor Business News, "Tegal says TEL Enjoined from selling etchers with dual-frequency technology, " http://seminews.com/sbn2/news/19999090a1. Shtml(Sept. 01,1999).
Zajac, J., et al., "Automated Plasma Etch Systems for VLSI, " Plasma Seminar Proceedings-1984 Tenth Annual (Tegal Corporation).
Zajac, John, et al., "Automated Plasma Etch Systems for VLSI, " Technical Program Proceedings, Semicon/ West 1984 (May 22-24, San Mateo, California), pp. i-v, 27-35.
Zajac, John, et al., "Multi-Electrode Plasma Etching, " Tegal Process Review (Tegal Corp., May 1984),.sub.1 (1):1-3.
Singer, Peter H., "The Challenge: Automating the Industry, " Technical Proceedings-Semion West '84 (Semiconductor International, May 1984) pp. 272-275.

ART-UNIT: 2172

PRIMARY-EXAMINER: Corrielus; Jean M.

ATTY-AGENT-FIRM: Townsend and Townsend and Crew LLP

ABSTRACT:

According to the invention, a database combining module, such as a software on a server, is used together with a resource manager software of a disk storage system in which a database is stored. A resource allocation for the disk storage system is determined to satisfy specifications requested by a user provided to a database combining module. Host paths and volumes are allocated and snapshots are controlled based on the processing of these software modules.

17 Claims, 7 Drawing figures

☐ 2. Document ID: US 6236997 B1

L8: Entry 2 of 6 File: USPT May 22, 2001

US-PAT-NO: 6236997
DOCUMENT-IDENTIFIER: US 6236997 B1

TITLE: Apparatus and method for accessing foreign databases in a heterogeneous database system

DATE-ISSUED: May 22, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Bodamer; Roger	Mountain View	CA		
Voss; Eric	Foster City	CA		
Draaijer; Jacco	Mountain View	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Oracle Corporation	Redwood Shores	CA			02

APPL-NO: 08/ 880333 [PALM]
DATE FILED: June 23, 1997

PARENT-CASE:
CROSS-REFERENCE TO RELATED APPLICATIONS This application is related to commonly-assigned, application Ser. No. 08/880,326, now U.S. Pat. No. 5,987,463 filed even date herewith, entitled "Apparatus and Method for Calling External Routines in a Database System," inventors Jacco Draaijer, Roger Bodamer, and Eric Voss, the disclosure of which is incorporated in its entirety herein by reference. This application is related to commonly-assigned, application Ser. No. 08/880,325, now U.S. Pat. No. 6,041,344 filed even date herewith, entitled "Apparatus and Method for Passing Statements to Foreign Databases by Using a Virtual Package," inventors Roger Bodamer, Jacco Draaijer, Eric Voss, Raghu Mani, the disclosure of which is incorporated in its entirety herein by reference. This application is related to commonly-assigned, application Ser. No. 08/880,327 now abandoned filed even date herewith, entitled "Apparatus and Method for Transparent Access of Foreign Databases in a Heterogeneous Database System," ,". inventors Roger Bodamer, Jacco Draaijer, Eric Voss, Raghu Mani, the disclosure of which is incorporated in its entirety herein by reference.

INT-CL: [07] G06 F 17/30

US-CL-ISSUED: 707/10; 709/201
US-CL-CURRENT: 707/10; 709/201

FIELD-OF-SEARCH: 707/102, 707/10, 707/103, 709/201

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4736321	April 1988	Brown et al.	709/300
4949255	August 1990	Gerth et al.	709/304
5218697	June 1993	Chung	709/230
5257366	October 1993	Adair et al.	707/4
5416917	May 1995	Adair et al.	707/203
5452450	September 1995	Delory	707/10
5455948	October 1995	Poole et al.	707/102
5524253	June 1996	Pham et al.	709/202
5539886	July 1996	Aldred et al.	709/304
5542078	July 1996	Martel et al.	707/101
5596744	January 1997	Dao et al.	707/10
5608874	March 1997	Ogawa et al.	395/20076
5617533	April 1997	Wu et al.	714/38
5627972	May 1997	Shear	709/246
5651111	July 1997	McKeeman et al.	714/38
5655116	August 1997	Kirk et al.	707/1
5680618	October 1997	Freund	707/200
5682535	October 1997	Knudsen	395/701
5706499	January 1998	Kleewein et al.	707/10
5710918	January 1998	Lagarde et al.	707/10
5713014	January 1998	Durflinger et al.	707/4

5721904	February 1998	Ito et al.	707/8
5745754	April 1998	Lagarde et al.	707/104
5764949	June 1998	Huang et al.	395/500
5768577	June 1998	Kleewein et al.	707/10
5768589	June 1998	Bradley et al.	395/684
5787452	July 1998	McKenna	707/536
5794234	August 1998	Church et al.	707/4
5806066	September 1998	Golshani et al.	707/100
5859972	January 1999	Subramaniam et al.	709/203
5987463	November 1999	Draaijer et al.	707/10
6041344	March 2000	Bodamer et al.	709/203

ART-UNIT: 217

PRIMARY-EXAMINER: Breene; John E.

ASSISTANT-EXAMINER: Robinson; Greta L.

ATTY-AGENT-FIRM: Ditthavong & Carlson, P.C.

ABSTRACT:

An apparatus and method for accessing foreign processes in a heterogeneous database environment includes a local database server having a heterogeneous services module to selectively send requests to the foreign processes based on their respective capabilities. A client application sending a statement to the local database server is checked by the local server process to determine if the statement includes a reference to a foreign database system, or whether the statement is registered as an external routine. The heterogeneous services module selectively outputs a request to an agent process executing in an address space separate from the local server process and in communication with the foreign database. The agent process performs all necessary interaction with the foreign database, including data type translation. The agent includes a conversion module that includes data type conversion routines, and which may obtain additional conversion routines via an Application Programming Interface (API) from a foreign database driver corresponding to the foreign database. The converted request is output from the agent process to the foreign database. Hence, the heterogeneous services module manages client statements involving foreign database systems having limited capabilities, as well as client statements having expressions unrecognizable by the local database server. Use of the agent process ensures that the integrity of the local server process is protected.

32 Claims, 9 Drawing figures

☐ 3. Document ID: US 6226649 B1

L8: Entry 3 of 6 File: USPT May 1, 2001

US-PAT-NO: 6226649
DOCUMENT-IDENTIFIER: US 6226649 B1

TITLE: Apparatus and method for transparent access of foreign databases in a heterogeneous database system

DATE-ISSUED: May 1, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Bodamer; Roger	Mountain View	CA		
Draaijer; Jacco	Mountain View	CA		
Voss; Eric	Foster City	CA		
Mani; Raghu	Mountain View	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Oracle Corporation	Redwood Shores	CA			02

APPL-NO: 09/ 390028 [PALM]
DATE FILED: September 3, 1999

PARENT-CASE:
CROSS-REFERENCE TO RELATED APPLICATIONS This application is a continuation of the commonly-assigned, application Ser. No. 08/880,327, filed Jun. 23, 1997, entitled "Apparatus And Method For Transparent Access Of Foreign Databases In A Heterogeneous Database System," inventors Roger Bodamer, Jacco Draaijer, Eric Voss, and Raghu Mani, the disclosure of which is incorporated in its entirety herein by reference, now abandoned. This application is related to commonly-assigned, application Ser. No. 08/880,326, filed Jun. 23, 1997, entitled "Apparatus and Method for Calling External Routines in a Database System," inventors Jacco Draaijer, Roger Bodamer, and Eric Voss, now U.S. Pat. No. 5,987,463, issued Nov. 16, 1999, the disclosure of which is incorporated in its entirety herein by reference. This application is related to commonly-assigned, application Ser. No. 08/880,333, filed Jun. 23, 1997, entitled "Apparatus and Method for Accessing Foreign Databases in a Heterogeneous Database System," inventors Roger Bodamer, Eric Voss, and Jacco Draaijer, the disclosure of which is incorporated in its entirety herein by reference, now pending. This application is related to commonly-assigned, application Ser. No. 08/880,325, filed Jun. 23, 1997, entitled "Apparatus and Method for Passing Statements to Foreign Databases by Using a Virtual Package," inventors Roger Bodamer, Jacco Draaijer, Eric Voss, Raghu Mani, now U.S. Pat. No. 6,041,344, issued Mar. 21, 2000, the disclosure of which is incorporated in its entirety herein by reference.

INT-CL: [07] G06 F 17/30

US-CL-ISSUED: 707/104; 707/2, 707/3, 707/102, 707/104, 707/10, 709/203, 709/217, 709/245, 709/246
US-CL-CURRENT: 707/104.1; 707/10, 707/102, 707/2, 707/3, 709/203, 709/217, 709/245, 709/246

FIELD-OF-SEARCH: 707/10, 707/2, 707/100, 707/3, 707/103, 707/104, 707/102, 709/203, 709/217, 709/245, 709/246

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4736321	April 1988	Brown et al.	709/313
4949255	August 1990	Gerth et al.	709/315
5218697	June 1993	Chung	709/230
5257366	October 1993	Adair et al.	707/4
5416917	May 1995	Adair et al.	707/203
5452450	September 1995	Delory	707/10
5455948	October 1995	Poole et al.	707/102
5524253	June 1996	Pham et al.	709/202

5539886	July 1996	Aldred et al.	709/318
5542078	July 1996	Martel et al.	707/101
5596744	January 1997	Dao et al.	707/10
5608874	March 1997	Ogawa et al.	709/246
5617533	April 1997	Wu et al.	714/38
5627972	May 1997	Shear	709/203
5651111	July 1997	McKeeman et al.	714/38
5655116	August 1997	Kirk et al.	707/1
5706499	January 1998	Kleewein et al.	707/10
5710918	January 1998	Lagarde et al.	707/10
5713014	January 1998	Durflinger et al.	707/4
5721904	February 1998	Ito et al.	707/8
5745754	April 1998	Lagarde et al.	707/104
5764949	June 1998	Huang et al.	707/102
5768577	June 1998	Kleewein et al.	707/10
5768589	June 1998	Bradley et al.	709/304
5787452	July 1998	McKenna	707/536
5794234	August 1998	Church et al.	707/4
5806066	September 1998	Golshani et al.	707/100
5859972	January 1999	Subramaniam et al.	709/246
5930793	July 1999	Kleewein et al.	707/10
5943671	August 1999	Kleewein et al.	707/9
5966707	October 1999	Van Huben et al.	707/10
5987463	November 1999	Draaijer et al.	707/10
6041344	March 2000	Bodamer et al.	709/203

OTHER PUBLICATIONS

Hsu et al., "Information resources management in heterogeneous, distributed environments:metadatabase approach", IEEE, pp. 604-625, Jan. 1991.*
Hsu et al., "A metadata system for information modeling and integration", IEEE, pp. 616-624, Jan. 1990.*
Kang et al., "An integrated access control in heterogeneous distributed database systems", IEEE, pp. 222-226, Nov. 1992.

ART-UNIT: 271

PRIMARY-EXAMINER: Alam; Hosain T.

ASSISTANT-EXAMINER: Corrielus; Jean M.

ATTY-AGENT-FIRM: Ditthavong & Carlson, P.C.

ABSTRACT:

An apparatus and method for accessing foreign processes in a heterogeneous database environment includes a local database server having heterogeneous services to selectively send requests to the foreign processes based on their respective capabilities. A client application sending a statement to the local database server is checked by the local server process to determine if the statement includes a reference to a foreign database system. The local server process selectively outputs a request to an agent process in communication with a foreign database via a generic Application Programming Interface (API). The request output to the foreign database is based on accessing a capabilities table specifying the operations that can be executed by the foreign database. Hence, the heterogeneous services within the local server process manages client statements involving foreign database systems having limited capabilities, and uses the

agent process to manage interactions with the foreign database systems, including converting data, to preserve the integrity of the local server process and provide the appearance to the client application of a homogeneous distributed database system.

23 Claims, 8 Drawing figures

☐ 4. Document ID: US 5566332 A

L8: Entry 4 of 6 File: USPT Oct 15, 1996

US-PAT-NO: 5566332
DOCUMENT-IDENTIFIER: US 5566332 A

TITLE: Method and combination for minimizing data conversions when data is transferred between a first database storing data in a first format and a second database storing data in a second format

DATE-ISSUED: October 15, 1996

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Adair; John G.	Austin	TX		
Coyle, Jr.; Daniel J.	Los Gatos	CA		
Grafe; Robert J.	Austin	TX		
Lindsay; Bruce G.	San Jose	CA		
Reinsch; Roger A.	Cupertino	CA		
Resch; Robert P.	Byron	MN		
Selinger; Patricia G.	San Jose	CA		
Zimowski; Melvin R.	San Jose	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 08/ 288837 [PALM]
DATE FILED: August 11, 1994

PARENT-CASE:
This application is a continuation of application Ser. No. 08/066,323, filed May 21, 1993, now U.S. Pat. No. 5,416,917 which is a continuation of Ser. No. 07/500,031 filed Mar. 27, 1990, now abandoned.

INT-CL: [06] G06 F 17/30, G06 F 15/163

US-CL-ISSUED: 395/600; 395/200.12, 395/200.18
US-CL-CURRENT: 707/101; 709/246

FIELD-OF-SEARCH: 395/600, 395/500, 395/200.09, 395/200.12, 395/200.18

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4677552	June 1987	Sibley, Jr.	364/408
4714995	December 1987	Materna et al.	395/600
4774655	September 1988	Kollin et al.	395/600
5058000	October 1991	Cox et al.	395/600
5109483	April 1992	Baratz et al.	395/200
5278978	January 1994	Demers et al.	395/600

OTHER PUBLICATIONS

Hu, "Making Ends Meet: Interconnecting Electronic Mail Networks", Data Communications, Sep. 1988, vol. 17, Is. 10 pp. 128-141.
Ruschitzka et al, "Heterogeneous Data Translations Based on Environmental Grammars", IEEE Transactions on Software Engineering, Oct. 1989, vol. 15 Is. 10 pp. 1236-1251.
Mantelman, "Apples and Unix and DOS oh my." Data Communications Feb. 1988; vol. 17 Is 2; pp. 69-77.
Strevell et al., "High Speed Transformation of Primitive Data Types In a Heterogeneous Distributed Computer System," Distributed Computing Systems 1988 Internat. Conf. pp. 41-46.

ART-UNIT: 237

PRIMARY-EXAMINER: Black; Thomas G.

ASSISTANT-EXAMINER: Choules; Jack M.

ATTY-AGENT-FIRM: Baker, Maxham, Jester & Meador

ABSTRACT:

The invention establishes the context in which data exchanged between dissimilar relational database management systems can be mutually understood and preserved, and data conversions can be minimized. The invention accomplishes this by establishing layers of descriptive information which isolate machine characteristics, levels of support software, and user data descriptions. Optimized processing is achieved by processing the different descriptor levels at different times during the development and execution of the database management systems. Minimal descriptive information is exchanged between the cooperating database management systems. Any data conversions that may be necessary are done only by the receiver of the data, and only at the point where it is necessary to have the data represented in the receiver's native format for processing. For sending and receiving systems which match, data conversion is completely avoided, even when the data may have been forwarded through intermediate DBMS systems which do not match. For sending and receiving systems which do not match, data conversion is minimized. The data conversion routines and tables in each system are also minimized by requiring conversion only INTO a system's native format, never FROM its native format into some other format.

19 Claims, 13 Drawing figures

☐ 5. Document ID: US 5446880 A

L8: Entry 5 of 6 File: USPT Aug 29, 1995

US-PAT-NO: 5446880
DOCUMENT-IDENTIFIER: US 5446880 A

TITLE: Database communication system that provides automatic format translation and transmission of records when the owner identified for the record is changed

DATE-ISSUED: August 29, 1995

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Balgeman; Timothy E.	Warrenville	IL		
Clodi; Diane M.	Oswego	IL		
Haddleton, Jr.; Robert W.	Naperville	IL		
North; John R.	Geneva	IL		
Selby; Judith A.	Wheaton	IL		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
AT&T Corp.	Murray Hill	NJ			02

APPL-NO: 07/ 937814 [PALM]
DATE FILED: August 31, 1992

INT-CL: [06] G06 F 17/30

US-CL-ISSUED: 395/600; 395/200.01, 364/241.7, 364/242.5, 364/284.3, 364/DIG.1
US-CL-CURRENT: 707/9; 707/201

FIELD-OF-SEARCH: 395/600, 395/200, 340/825.05, 364/DIG.1, 364/DIG.2, 364/241.7, 364/241.8, 364/242.5, 364/384.3

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4031512	June 1977	Fabor	340/825.05
4135240	January 1979	Ritchie	395/600
4714995	December 1987	Materna et al.	395/600
4714995	December 1987	Materna et al.	395/600
4933846	June 1990	Humphrey et al.	395/325
5119465	June 1992	Jack et al.	395/500
5249231	September 1993	Covey et al.	395/425
5276869	January 1994	Forrest et al.	395/600
5283887	February 1994	Zachery	395/500
5329618	July 1994	Moati et al.	395/200

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0216535A2	January 1987	EP	

OTHER PUBLICATIONS

Papasoglou et al. "Distributed Heterogeneous Information Systems & Schema Transformation",

Parbase-90, Conf Date, 7-9 Mar. 1990, pp. 388-397 IEEE.
Pu, "Superdatabases for Composition of Heterogeneous Databases", Pr. 4th Int. Conf. On Data Engineering, date 1-5 Feb. 1988; pp. 548-555.
Chung, "DATAPLEX: an access to Heterogeneous distributed databases", Communications Of The ACM, Jan. 1990: V33 issue h1, p. 70-81.
J. Gray & S. Metz "Solving The Problems of Distributed Databases", Data Communications, vol. 12, No. 10, Oct., 1983, pp. 183-192.
R. Ahmed et al. "The Pegasus Heterogeneous Multidatabase Systems", Computer, vol. 24, No. 12, Dec., 1991, pp. 19-26.
"Financial Systems Interface", IBM Technical Disclosure Bulletin, vol. 31, No. 5, Oct., 1988, pp. 36-39.

ART-UNIT: 237

PRIMARY-EXAMINER: Black; Thomas G.

ASSISTANT-EXAMINER: Choules; Jack M.

ATTY-AGENT-FIRM: Warren; Charles L.

ABSTRACT:

A database interface and associated communication program are installed in each node in a communication network. The database interface provides a translation from the record format of the database and a standardized format for transmission to other nodes thus providing translation between the formats of the different databases formats as records are transmitted between the databases on different nodes. The communication program transmits records that have been translated to the standardized format to nodes which contain corresponding records maintaining current records at each node. A node having a corresponding node can assign an owner. Assigning an owner causes the record to be sent to the database that is associated with the new owner. The owner has the responsibility for acting on the record. The originator of the record is also identified.

15 Claims, 9 Drawing figures

☐ 6. Document ID: US 5416917 A

L8: Entry 6 of 6 File: USPT May 16, 1995

US-PAT-NO: 5416917
DOCUMENT-IDENTIFIER: US 5416917 A
**** See image for Certificate of Correction ****

TITLE: Heterogenous database communication system in which communicating systems identify themselves and convert any requests/responses into their own data format

DATE-ISSUED: May 16, 1995

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Adair; John G.	Austin	TX		
Coyle, Jr.; Daniel J.	San Jose	CA		
Grafe; Robert J.	Austin	TX		
Lindsay; Bruce G.	San Jose	CA		
Reinsch; Roger A.	Cupertino	CA		

Resch; Robert P.	Byron	MN
Selinger; Patricia G.	San Jose	CA
Zimowski; Melvin R.	San Jose	CA

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 08/ 066323 [PALM]
DATE FILED: May 21, 1993

PARENT-CASE:
Cross Reference to Pending Applications This application includes material which overlaps material in pending U.S. Pat. application Ser. No. 499,114, filed Mar. 26, 1990, entitled "METHOD AND APPARATUS FOR DESCRIBING DATA TO BE EXCHANGED BETWEEN PROCESSES", now U.S. Pat. No. 5,278,978, inventors; Richard Demers, et al.; and U.S. Pat. application Ser. No. 500,032, filed Mar. 27, 1990, it, entitled "QUERY LANGUAGE EXECUTION ON HETEROGENEOUS DATABASE SERVERS", now U.S. Pat. No. 5,257,366, inventors: John G. Adair, et al. This is a continuation of application Ser. No. 07/500,031 filed Mar. 27, 1990, now abandoned.

INT-CL: [06] G06 F 3/00, G06 F 13/00

US-CL-ISSUED: 395/500; 395/600, 395/200, 364/239.3, 364/260.4, 364/282.1, 364/DIG.1
US-CL-CURRENT: 707/203; 707/10

FIELD-OF-SEARCH: 395/800, 395/600, 395/200, 395/325, 395/500, 370/60

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
T940012	November 1975	Beretvas et al.	
4455605	June 1984	Cormier et al.	364/200
4574350	March 1986	Starr	364/200
4604710	August 1986	Amezcua et al.	364/900
4648061	March 1987	Foster	395/200
4677552	June 1987	Sibley, Jr.	364/408
4714995	December 1987	Materna et al.	364/200
4755929	July 1988	Outous, et al.	364/200
4774655	September 1988	Kollin et al.	364/200
4788657	November 1988	Douglas et al.	364/900
4803643	February 1989	Hickey	364/523
4845658	July 1989	Gifford	364/900
4868866	September 1989	Williams, Jr.	380/49
5058000	October 1991	Cox et al.	364/200
5109483	April 1992	Baratz et al.	395/200

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
2575842	July 1986	FR	

2612661	September 1988	FR
1111176	August 1984	SU

OTHER PUBLICATIONS

Chin-Wan Chung, "DATAPLEX: An Access to Heterogeneous Distributed Databases," in Communications of the ACM, Jan. 1990, vol. 33, No. 1, New York, New York, pp. 70-80.
Informationstechnik It, vol. 29, No. 3, 1987, the whole document, Effelsberg, "Database Access In Networks".
"DATAPLEX: An Access to Heterogeneous Distributed Databases", pp. 71, 72, and 75, Chung, Jan. 1990.
1988 International Zurich Seminar on Digital Communications, 8 Mar. 1988, pp. 253-259, IEEE.
Ishizaki, Jirou--Translation Summary of Japanese Application No. 58-165161, "Simple Data Base System", Sep. 30, 1983.
Philibert, J. P. "Data Management System for Data Base Conversion," IBM TDB, vol. 15 No. 8, Jan. 1973, pp. 2463-2464.
Edstrom et al., "Interconnector for Diverse Types of Program Apparatus," IBM TDB, vol. 18 No. 8, Jan. 1976, pp. 2415-2417.

ART-UNIT: 237

PRIMARY-EXAMINER: Lee; Thomas C.

ASSISTANT-EXAMINER: Harrity; Paul

ATTY-AGENT-FIRM: Baker, Maxham, Jester & Meador

ABSTRACT:

The invention establishes the context in which data exchanged between dissimilar relational database management systems can be mutually understood and preserved, and data conversions can be minimized. The invention accomplishes this by establishing layers of descriptive information which isolate machine characteristics, levels of support software, and user data descriptions. Optimized processing is achieved by processing the different descriptor levels at different times during the development and execution of the database management systems. Minimal descriptive information is exchanged between the cooperating database management systems. Any data conversions that may be necessary are done only by the receiver of the data, and only at the point where it is necessary to have the data represented in the receiver's native format for processing. For sending and receiving systems which match, data conversion is completely avoided, even when the data may have been forwarded through intermediate DBMS systems which do not match. For sending and receiving systems which do not match, data conversion is minimized. The data conversion routines and tables in each system are also minimized by requiring conversion only INTO a system's native format, never FROM its native format into some other format.

9 Claims, 13 Drawing figures

Term	Documents
SECOND	2447495
SECONDS	336180
FORMAT	184594

FORMATS	63715
(7 AND (FORMAT NEAR SECOND)).USPT.	6
(L7 AND (SECOND NEAR FORMAT)).USPT.	6

Display Format: FRO Change Format

Previous Page Next Page Go to Doc#

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 2 of 2 returned.

☐ 1. Document ID: US 6343275 B1

L3: Entry 1 of 2 File: USPT Jan 29, 2002

US-PAT-NO: 6343275
DOCUMENT-IDENTIFIER: US 6343275 B1

TITLE: Integrated business-to-business web commerce and business automation system

DATE-ISSUED: January 29, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Wong; Charles	Los Altos Hills	CA	94022	

APPL-NO: 09/ 356327 [PALM]
DATE FILED: July 16, 1999

PARENT-CASE:
This application is a continuation, of application Ser. No. 08/995,591, filed Dec. 22, 1997 now U.S. Pat. No. 6,115,690.

INT-CL: [07] G06 F 17/60

US-CL-ISSUED: 705/26; 705/1, 705/7, 705/8, 705/9, 707/1, 707/10, 707/100, 707/102, 707/523, 707/217, 709/201
US-CL-CURRENT: 705/26; 705/1, 705/7, 705/8, 705/9, 707/1, 707/10, 707/100, 707/102, 709/201, 715/523

FIELD-OF-SEARCH: 705/1, 705/7, 705/8, 705/9, 705/26, 707/1, 707/10, 707/100, 707/102, 707/217, 707/523, 709/201

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4882675	November 1989	Nichtberger et al.	705/14
5237497	August 1993	Sitarski	705/8
5311438	May 1994	Sellers et al.	700/96
5353218	October 1994	De Lapa et al.	705/14
5913061	June 1999	Grupta et al.	709/310
5968110	October 1999	Westrope et al.	705/27
5991739	November 1999	Cupps et al.	705/26

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
996273	October 1999	EP	

OTHER PUBLICATIONS

Business to Business on the Internet: Using the web to cut costs and build sales, Computer Reseller news pp 34, Nov. 1996.*
dialog reference file 9 00960974, Eric Clemons, Segmentation, differentiation, and flexible pricing: Experience with information technology and segment-tailored strategies, Journal of Management Infrmation Systems: JMIS PP 9-36, 1994.*
dialog reference file 9 00960974, Eric Clemons, Segmentation, differentiation, and flexible pricing: Experience with information technology and segment-tailored strategies, Journal of Management Infrmation Systems: JMIS PP 9-36.

ART-UNIT: 2162

PRIMARY-EXAMINER: Stamber; Eric W.

ASSISTANT-EXAMINER: Alvarez; Raquel

ABSTRACT:

A software system business-to-business Web commerce (Web business, or e-business) and automates to the greatest degree possible, in a unified and synergistic fashion and using best proven business practices, the various aspects of running a successful and profitable business. Web business and business automation are both greatly facilitated using a computing model based on a single integrated database management system (DBMS) that is either Web-enabled or provided with a Web front-end. The Web provides a window into a "seamless" end-to-end internal business process. The effect of such integration on the business cycle is profound, allowing the sale of virtually anything in a transactional context (goods, services, insurance, subscriptions, etc.) to be drastically streamlined.

19 Claims, 339 Drawing figures

☐ 2. Document ID: US 6115690 A

L3: Entry 2 of 2 File: USPT Sep 5, 2000

US-PAT-NO: 6115690
DOCUMENT-IDENTIFIER: US 6115690 A

TITLE: Integrated business-to-business Web commerce and business automation system

DATE-ISSUED: September 5, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Wong; Charles	Los Altos Hills	CA	94022	

APPL-NO: 08/ 995591 [PALM]
DATE FILED: December 22, 1997

INT-CL: [07] G06 F 17/60

US-CL-ISSUED: 705/7; 705/1, 705/8, 705/30, 705/34, 364/709.06, 364/479.07
US-CL-CURRENT: 705/7; 700/237, 705/1, 705/30, 705/34, 705/8, 708/136

FIELD-OF-SEARCH: 235/380, 364/468.02, 364/468.14, 364/468.21, 364/479.06, 364/479.07, 364/479.08, 364/705.06, 364/709.06, 705/34, 705/1, 705/30, 705/7, 705/8

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5101352	March 1992	Rembert	705/8
5191522	March 1993	Bosco et al.	705/4
5224034	June 1993	Katz et al.	705/7
5311438	May 1994	Sellers et al.	364/468.02
5450317	September 1995	Lu et al.	705/10
5528490	June 1996	Hill	395/712
5557515	September 1996	Abbruzzese et al.	705/9
5592378	January 1997	Cameron et al.	705/27
5596502	January 1997	Koski et al.	364/468.01
5615109	March 1997	Eder	705/8
5621201	April 1997	Langhans et al.	235/380
5638519	June 1997	Haluska	705/28
5666493	September 1997	Wojcik et al.	705/26

ART-UNIT: 271

PRIMARY-EXAMINER: Cosimano; Edward R.

ASSISTANT-EXAMINER: Alvarez; Raquel

ATTY-AGENT-FIRM: Burns, Doane, Swecker & Mathis, LLP

ABSTRACT:

A software system business-to-business Web commerce (Web business, or e-business) and automates to the greatest degree possible, in a unified and synergistic fashion and using best proven business practices, the various aspects of running a successful and profitable business. Web business and business automation are both greatly facilitated using a computing model based on a single integrated database management system (DBMS) that is either Web-enabled or provided with a Web front-end. The Web provides a window into a "seamless" end-to-end internal business process. The effect of such integration on the business cycle is profound, allowing the sale of virtually anything in a transactional context (goods, services, insurance, subscriptions, etc.) to be drastically streamlined.

85 Claims, 129 Drawing figures

Full | Title | Citation | Front | Review | Classification | Date

L8: Entry 6 of 6 File: USPT May 16, 1995

DOCUMENT-IDENTIFIER: US 5416917 A

**** See image for Certificate of Correction ****

TITLE: Heterogenous database communication system in which communicating systems identify themselves and convert any requests/responses into their own data format

Parent Case Text (2):

This application includes material which overlaps material in pending U.S. Pat. application Ser. No. 499,114, filed Mar. 26, 1990, entitled "METHOD AND APPARATUS FOR DESCRIBING DATA TO BE EXCHANGED BETWEEN PROCESSES", now U.S. Pat. No. 5,278,978, inventors; Richard Demers, et al.; and U.S. Pat. application Ser. No. 500,032, filed Mar. 27, 1990, it, entitled "QUERY LANGUAGE EXECUTION ON HETEROGENEOUS DATABASE SERVERS", now U.S. Pat. No. 5,257,366, inventors: John G. Adair, et al. This is a continuation of application Ser. No. 07/500,031 filed Mar. 27, 1990, now abandoned.

Brief Summary Text (7):

The machines upon which the DBMSs of a heterogeneous database system run all represent information in different internal formats. For example, numeric information on PS/2 machines is stored with the bytes in low order to high order sequence. On other machines, such information may be stored in high order to low order sequence. For floating point information, there are IEEE floating point machines and hexadecimal floating point machines. Character information is processed in many different code representations, the choice of which reflects historical or cultural roots.

Brief Summary Text (11):

Provision is made in the prior art for solving the problems of machine and system incompatibility in a distributed, heterogeneous database system. Three solutions are of interest.

Brief Summary Text (12):

The earliest solution may be termed "application beware". This solution usually starts as a connection between identical database systems which grows over time to incorporate some machines which differ slightly from the original. In these solutions, there is no way for the system to automatically handle the differences, with the result that the application program was given this responsibility. If access to heterogeneous databases was needed badly enough, the application was written to make any necessary accommodations.

Brief Summary Text (15):

Thus, there is an evident need in distributed, heterogeneous database systems to support effective and accurate exchange of data, while reducing the number of conversions, and the communications overhead. It is also desirable to perform any needed conversion at the site where the data to be converted will be processed.

Brief Summary Text (18):

To minimize the number of conversions, a method is provided whereby no data is converted until it is needed in a specific format for processing. This method allows each system to always send all data in its preferred (native) format, or the current format of the data if the data is just passing through an intermediate system without being processed. When data is received, it is converted to the native format of the system only when it has reached its final destination (in being passed to the application, or stored in the database), or when it actually needs to be processed by an intermediate system.

Brief Summary Text (19):

According to the invention, a database in a distributed, heterogeneous database system contains predefined descriptions of all machine environments in the system (machine descriptors) and predefined descriptions of database language structures (system descriptors) for each DBMS with which it can perform data exchange. When a database operation begins which requires two heterogeneous databases to conduct an information exchange, a communication link is established between them. Next, each DBMS identifies its machine and system descriptors to the other. This establishes a data context and is done only once for the life of the communication link. Once established, requests, responses and data can be sent or received. When any data is sent, it is sent in the native format of the sender. Specific descriptions of the data precede the data itself and refer to the machine and system descriptors earlier identified. When the data is received, information contained in the specific descriptions enable these descriptions to be referenced to the machine and language descriptors and passed off to a conversion process in the receiver. Taken together, the specific descriptions, and the machine and system descriptors which they reference, precisely characterize the environment where the data originated and establish, at the receiver, a context for accurate conversion of the data into a format which is native to the machine/system combination of the receiving site.

Detailed Description Text (5):
Furthermore, there are many SQL identified standard control blocks, such as the SQL communication area (SQLCA) which is defined in terms of SQL types. The SQLCAs in the machine environments defined above are not identical. This invention formalizes a method and means for exchanging information between heterogeneous DBMSs about machine characteristics and DBMS language structures such that little or no descriptive information must be exchanged at run-time in order to convert data to a native form. Additionally, when database sites match, even though the data may have been forwarded through intermediate DBMS systems which do not match, no conversions are performed at all, thus preserving completely the integrity of the data being exchanged. When these sites do not match, data conversions are performed close to the ultimate point of use where errors introduced by imperfect conversions can be dealt with according to the needs of the requesting application.

Detailed Description Text (15):
The context for conversion also requires information describing characteristics of the language in which the DBMS sending the information is written. In this description it is assumed that all of the DBMS sites are written in one or another version of an SQL-based language. In order to accommodate non-identical representations of control information produced by these varying versions, a system-level language characteristic descriptor must be available to the converting machine in order to convert the language-specific portions of the user data. In FIG. 1A, the language-specific portions are the communication areas. In order to provide this system-level information about the language characteristics, the converting machine is provided a system-level information unit descriptor. In FIG. 1B, the system-level descriptor for the personal computer language environment is indicated by reference numeral 45.

Detailed Description Text (18):
Thus, in FIG. 1B, assuming that the mainframe machine 50 is sending data, its machine-level and system-level descriptors 54 and 55 must be made available to the requesting personal computer machine 40 and linked to application level descriptors 57 in order to convert the user data of FIG. 1A from the representation 20 to the representation 10. Relatedly, the descriptors 54, 55 and 57 must be made available to the machine 40. Similarly, for data transferred in the opposite direction, that is, from the personal computer machine 40 to the mainframe machine 50, the descriptors 44, 45 and 47 would have to be made available to the mainframe machine 50 in order to convert user data 10 into the form of user data 20 in FIG. 1A.

Detailed Description Text (19):
In the practice of this invention, it is asserted that, during the implementation of each DBMS site which participates in a distributed, heterogeneous database system, a decision is made as to which specific machine representations will be supported at the site. It is asserted that the "native" representation of the machine on which the site's DBMS will run is supported. That is, if data from another DBMS identical to the receiving site's DBMS is received, the representation is "native" and requires no conversion. Additionally, other machine types will be supported as partners. Generally, the "non-native" representations of data types will be converted to "native" ones at the receiving site before processing. In the preferred embodiment, at each site, there is maintained a list of acceptable machine partner types at each site. Thus, at the site of the personal computer machine 40, a list of acceptable machine

types includes the machine and code page corresponding to identifiers 51 and 52 in the machine 50. The list indexes to a set of descriptors which include the machine-level and system-level descriptors for all acceptable system partners. Thus, if the personal computer machine 40 receives simply identification of the machine and language present at the site 50, these identifications can be brought to the list which will index to the necessary descriptors in the list of descriptors, the indexed descriptors corresponding to the descriptors 54 and 55.

Detailed Description Text (22):
Upon receipt, the receiver uses the descriptors obtained in response to the machine and system-level identifier sent by the machine 50 and the references contained in the application level descriptor 57 to convert the user data 20 from the server's representations to the receiver's representations for subsequent processing.

Detailed Description Text (39):
Refer now to FIG. 4 for an understanding of the structure and content of a user data descriptor and how its linkages with machine- and system-level descriptors provide a complete context which enables a receiver DBMS to understand and convert user data from non-native to native form. The user data descriptor 200 provides a system-level and machine-level-independent way to describe user data. It is contemplated that this descriptor would not be built until run time, since, particulars of user data aren't known until after installation of a DBMS and creation and population of the relational tables in the DBMS, and until receipt of a particular request for data.

Detailed Description Text (45):
FIG. 5 illustrates a procedure for establishing a conversion context using the data objects whose structures and functions have been described above. The procedure includes generic commands for requesting and completing a communication connection and for requesting and providing data. The procedure of FIG. 5 is illustrated as a simplified sequence which shows only the parameters which must, of necessity, be exchanged for transferring data between requesting and serving machines and establishing the translation context in the receiving machine for conversion of the data. It is reiterated that the procedure of the invention is not concerned with how data is converted, but rather with where the data is converted. Particularly, the invention enables the receiver of the data to convert the data for establishing particular values for those parameters.

Detailed Description Text (46):
The structure of FIG. 5 places all of the requesting machine actions in the left-hand side of the drawing and all of the serving machine actions in the right-hand side. Thus, in step 240 a user at the requesting machine first issues a request for data which is maintained by a DBMS in the serving machine. Next, in step 252 the requesting machine executes a REQUEST-CONNECTION command sequence directed to the serving machine. The REQUEST-CONNECTION sequence includes parameters to identify the requesting machine, its character code, and its language level. In the example, the requesting machine is identified as a DBMS executing on a PS/2 personal computer, utilizing a specific code page (437) for character coding, and operating at a specific DBMS language level (SQLAM3). The parameters in the REQUEST-CONNECTION step are no more than identifiers. Next, in step 254 the serving machine receives the connection request, validates the machine- and system-level identifiers, and uses these identifications to index to machine- and system-level descriptors. Assuming that the serving machine recognizes and possesses the machine- and system-level descriptors identified in the connection request, it executes a COMPLETE-CONNECTION in step 256 including parameters which provide machine- and system-level identification to the requesting machine. In step 258, the COMPLETE-CONNECTION response received from the serving machine is validated in the requesting machine and the identifiers are used to index the appropriate the machine- and system-level descriptors. For example, in the example of FIG. 5, the requesting machine would index to the machine and system descriptors 70 and 100 illustrated in FIG. 2B, 3B, and 4A. Next, in step 260, the requesting machine assembles a data request and sends it in step 263 as a REQUEST-DATA command to the serving machine. This REQUESTDATA optionally includes an input data descriptor (built in step 260) with optional input data, if appropriate for the SQL to be executed. In step 265, the serving machine receives the REQUEST-DATA command, including optional input data and descriptor. If input data is present, a conversion process is called in step 267. The serving machine responds to the REQUESTDATA command in step 269 by obtaining the requested user data and building a user data descriptor such as the descriptor 200 illustrated in FIG. 4A, and executes a PROVIDE-DATA command in step 270 by sending the user-data descriptor and the user

data as, for example, the fast row in the user data 20 in FIGS. 1B and 4B. The requesting machine then, in step 272, receives the transmitted descriptor and user data .and calls a conversion process to convert the data. According to the needs of a particular implementation, the bindings between the user data descriptor and the indexed machine and system descriptors can be done in step 272 or left for the called conversion process. In step 272, a second REQUEST-DATA command is sent, its parameters are processed, with the server, in step 276, obtaining and returning the requested user data with another PROVIDEDATA command (step 278). The requesting machine again, in step 279, processes the data according to the user data descriptor received with the first PROVIDEDATA command, and to the system level, machine type, and code page values received during the connection portion of the process. Now if the receiver requires more data, it issues another REQUEST-DATA command, with the server returning the data with another PROVIDE-DATA command, and so on.

Detailed Description Text (50):
Having verified that the receiver's environment can be handled, the application manager 318 constructs a COMPLETE-CONNECTION command as illustrated in FIG. 5. The command states the serving machine's characteristics in the MACH code page parameters and the level of system function which the serving machine will support in the DBMS parameter. It is contemplated in the invention that the connection steps may repeat in order to support negotiation to a different system level than that originally requested. Next, the COMPLETE-CONNECTION response is returned to the requesting machine 280 by sending it on 328 to the agent 314 which communicates it on the communication link 312 to the receiver agent 310. The receiver agent 310 sends the COMPLETE-CONNECTION command and parameters at 330 to the application manager 306. In a manner similar to the manager 318, the manager 306 accesses the system and machine-level descriptors which describe the server machine and system characteristics from the dictionaries 342 and 346. Having validated that connection has been established, and having obtained the machine- and system-level descriptors at both ends of the connection, processing continues. The requester's application manager 306 constructs a REQUEST-DATA command corresponding to the OPEN request from the application program 300. This may include an optimal input data descriptor and input data. This command is sent to the server's application manager 3 18 via 308, 310, 312, 314 and 316. The application manager 318 recognizes a "first request", processes any input data descriptor, converts any input data, and issues an OPEN command 332 to the local DBMS 334, which returns the status in the form of an SQL communication area 336. Next, the manager 318 issues a request on 332 to determine the format of the answer set. The DBMS 334 builds an SQL data area describing the rows which will be returned. This data area is returned on 336 to the server application manager 318 which constructs the user data descriptor for the data. Assuming that data is buffered into the manager 318 and that there is room in the buffer, the server's application manager 318 issues an SQL FETCH request on 332 to the DBMS 334 which returns a first row of data on 336. (It is assumed that the data to be returned corresponds to the user data 20 in FIGS. 1A and 4B.) Next, the server application manager 318 places the data in a reply buffer and issues a PROVIDE-DATA command to send the data to the requester's application manager 306 via 328, 314, 312, 310, and 330. It is contemplated, but not required, that the server's application manager 318 may read ahead to fill buffers in anticipation of the next REQUEST-DATA command.

Detailed Description Text (51):
Upon receiving data with the first PROVIDE-DATA command, the receiver's application manager 306 processes the user data descriptor to verify correctness and to prepare to convert data subsequently received. Assuming no errors are found, the application manager 306 then sends the result of the OPEN back to the application program 300 on 338. From now on, the receiver's application manager 306 may, but is not required to, read ahead requesting additional buffers from the server's application manager 318 in anticipation of FETCH requests.

Detailed Description Text (52):
Having successfully opened a Cursor, the application program 300 issues an SQL FETCH which is processed by the receiver's application manager 306. At this time, the receiver's application manager 306 has a row of data to return, and converts the data and returns it on 338 to the application program 300. The application 300 processes the data and SQL communication area received and subsequently issues another FETCH command on 304 to obtain another row of data. The requester's application manager 306 having no data to satisfy this request sends another REQUEST-DATA command to the server's application manager 318 via 308, 310, 312, 314, and 316. The server's application manager 318 having read ahead, has a buffer containing the last two rows of data. These rows are sent with the PROVIDE-DATA command to the requester's application

manager 306 via 328, 314, 312, 310, and 330. The application manager 306 then converts the first row of the buffer using the previously constructed conversion descriptors and passes it on 338 to the application 300.

Detailed Description Text (53):
To complete the example, the application 300 issues another SQL FETCH request on 304 to request the last row of the answer set. The receiver's application manager 306 converts this last row of data and returns it to the application.

Detailed Description Text (64):
A data converter is not illustrated as an element in any of the figures, it being understood, that data format conversion is well-known in the art. For example, U.S. Pat. No. 4,559,614 of Peek et al., assigned to the Assignee of this application, describes in detail how conversion from a first to a second internal code format is accomplished for data transmitted between two dissimilar computer systems. To the extent necessary, this patent is incorporated herein by reference.

Current US Original Classification (1):
707/203

Current US Cross Reference Classification (1):
707/10

CLAIMS:

1. In a system including a first database system for managing a first database including data having a first data format and a second database system for managing a second database including data having a second data format, a method for converting data transmitted between the first and second database systems, the method including the steps of:

storing at the first database system and at the second database system respective sets of machine descriptors describing machine data formats and character code sets, and system descriptors describing system language characteristics;

sending a request for connection from the first database system to the second database system, the request for connection including information identifying a first machine on which the first database system executes, a character code used by the first machine, and a system language used by the first database system;

in response to the request for connection:

validating a first machine descriptor and a first system descriptor stored at the second database system, the first machine descriptor describing a machine data format and a character code used by the first machine and the first system descriptor describing a system language used by the first database system; and

sending a connection response from the second database system to the first database system, the connection response including information identifying a second machine on which the second database system executes, a character code used by the second machine, and a system language used by the second database system;

in response to the connection response, validating a second machine descriptor and a second system descriptor stored at the first database system, the second machine descriptor describing a machine data format and character code used by the second machine and the second system descriptor describing a system language used by the second database system;

sending from the first database system to the second database system a database query command containing data in the first data format;

at the second database system, convening the data in the database query command into the second data format using the validated first machine descriptor and first system descriptor;

at the second database system, obtaining resulting data from the second database in response to

the database query command and the data in the second data format;

transmitting the resulting data to the first database system without convening the resulting data; and

at the first database system, convening the resulting data from the second data format to the first data format using the second machine descriptor and second system descriptor.

2. The method of claim 1, wherein the system further includes a first database system unit connected between the first database system and the second database system, wherein:

the step of sending the database query command includes receiving the database query command at the first database system unit and sending the database query command from the first database system unit to the second database system without converting the data in the database query command into a data format native to the first database system unit.

3. The method of claim 2, wherein the system further includes a second database system unit connected between the first database system and the second database system, wherein:

the step of transmitting the resulting data includes receiving the resulting data at the second database system unit and sending the resulting data from the second database system unit to the first database system without converting the data format of the resulting data at the second database system unit.

5. The method of claim 1, wherein the first database system is for executing on a first digital computer which represents data in a first internal format and the second database system is for executing on a second digital computer which represents data in a second internal format, the first and second internal formats being non-equivalent.

6. In a distributed database system including a plurality of database managers, each database manager for executing in a respective digital computer, a method for minimizing conversion of data transferred between database managers of the plurality of database managers, comprising the steps of:

storing at a first database manager and at a second database manager respective sets of machine descriptors describing machine data formats and character code sets, and system descriptors describing system language characteristics;

sending a request for connection from the first database manager to the second database manager, the request for connection including information identifying a first machine on which the first database manager executes, a character code used by the first machine, and a system language used by the first database manager;

in response to the request for connection:

validating a first machine descriptor and a first system descriptor stored at the second database manager, the first machine descriptor describing a machine data format and a character code used by the first machine and the first system descriptor describing a system language used by the first database manager; and

sending a connection response from the second database manager to the first database manager, the connection response including information identifying a second machine on which the second database manager executes, a character code used by the second machine, and a system language used by the second database manager;

in response to the connection response, validating a second machine descriptor and a second system descriptor stored at the first database manager, the second machine descriptor describing a machine data format and character code used by the second machine and the second system descriptor describing a system language used by the second database manager;

sending a database query command containing input data from the first database manager to the second database manager, the input data having a first data format used by the first database manager;

at the second database manager, if the second database manager uses a data format which is non-equivalent to the first data format used by the first database manager, using the validated first machine descriptor and first system descriptor to convert the input data from the first data format into a second data format used by the second database manager, otherwise, leaving the input data in the first data format;

at the second database manager, obtaining resulting data from a database in the data format used by the second database manager;

sending the resulting data in the data format used by the second database manager to the first database manager; and

at the first data base manager, if the data format used by the first database manager is not equivalent to the data format used by the second database manager, using the validated second machine descriptor and second system [language]descriptor to convert the resulting data into the first data format used by the first database manager.

8. In a system including a processor system for processing data having a first data format and a database system for managing a database including data having a second data format, a combination for convening data transmitted between the processor system and the database system, the combination including:

processor system storage storing a set of machine descriptors describing machine data formats, a character code set, and a set of system descriptors describing system language characteristics;

database system storage storing a set of machine descriptors describing machine data formats, a character code set, and a set of system descriptors describing system language characteristics;

a communications link connecting the processor system and the database system for communication;

means in the processor system for sending a request for connection from the processor system to the database system, the request for connection including information identifying a first machine on which the processor system executes, a character code used by the first machine, and a system language used by the processor system;

means in the database system responsive to the request for connection for validating a first machine descriptor and a first system descriptor stored in the database system storage, the first machine descriptor describing a machine data format and a character code used by the first machine and the first system descriptor describing a system language used by the processor system;

means in the database system for sending a connection response from the database system to the processor system, the connection response including information identifying a second machine on which the database system executes, a character code used by the second machine, and a system language used by the database system;

means in the processor system responsive to the connection response for validating a second machine descriptor and a second system descriptor stored in the processor system storage, the second machine descriptor describing a machine data format and a character code used by the second machine and a second system descriptor describing a system language used by the database system;

means in the processor system responsive to the connection response for sending a database query command containing data in the first data format from the processor system to the database system means in the database system for:

converting the database query command into the second data format using the validated first machine descriptor and first system descriptor;

obtaining resulting data from the database in response to the database query command and the data in the second format; and

transmitting the resulting data to the processor system without converting the resulting data; and

means in the processor system for converting the resulting data from the second data format to the first data format using the validated second machine descriptor and second system descriptor

9. The combination of claim 8, further including:

a router machine connected in the communications link between the processor system and the database system, the router machine including:

means for receiving the database query command;

means for determining that no conversion of the data in the database query command is necessary at the router machine; and

means for sending the database query command to the database system without converting the data in the database query command.

Previous Doc Next Doc Go to Doc#